IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:	§	Group Art Unit: 2181
Mitchell Alsup	§	
Gregory W. Smaus	§	Examiner: Geib, Benjamin P.
	§	
	§	Atty. Dkt. No.: 5500-81600/TT4899
	§	
Serial No. 10/614,970	§	
	§	
	§	
Filed: July 8, 2003	§	
	§	
For: System and Method of Implementing Microcode Operations as Subroutines		

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir/Madam:

Further to the Notice of Appeal filed October 18, 2006, Appellants present this Appeal Brief. Appellants respectfully request that the Board of Patent Appeals and Interferences consider this appeal.

I. **REAL PARTY IN INTEREST**

As evidenced by the assignment recorded at Reel/Frame 014299/0555, the subject application is owned by Advanced Micro Devices, Inc., a corporation organized and existing under and by virtue of the laws of the State of Delaware, and now having its principal place of business at One AMD Place, P.O. Box 3453, Sunnyvale, CA 94088.

II. RELATED APPEALS AND INTERFERENCES

No other appeals, interferences or judicial proceedings are known which would be related to, directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1-41 are pending and stand finally rejected. The rejection of claims 1-41 is being appealed. A copy of claims 1-41 is included in the Claims Appendix herein below.

IV. STATUS OF AMENDMENTS

No amendments to the claims have been submitted subsequent to the final rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 1 is directed to a microprocessor. (*See, e.g.,* FIG. 1, element 100; and p. 11, lines 3-4). The microprocessor includes a dispatch unit configured to dispatch operations. (*See, e.g.,* FIG. 1, element 104; and p. 12, line 14.)

The microprocessor also includes a scheduler coupled to the dispatch unit and configured to schedule dispatched operations for execution. (*See, e.g.,* FIG. 1, element 118; and p. 12, lines 15-16.)

In response to receiving a microcoded instruction, the dispatch unit is configured to dispatch to the scheduler a microcode subroutine call operation that includes a tag identifying a microcode subroutine associated with the microcoded instruction. (*See, e.g.,* FIG. 13C; p. 41 lines 23-28, and p. 42, lines 6-9.)

Independent claim 17 is directed to a computer system. (*See, e.g.,* FIG. 15.) The computer system includes a system memory. (*See, e.g.,* FIG. 1, element 200; and FIG. 15, element 404.) The computer system also includes a microprocessor coupled to the system memory. (*See, e.g.,* FIG. 1, element 100; and FIG. 15, element 10).

The microprocessor is similar to that described above regarding claim 1. Specifically, the microprocessor includes a dispatch unit configured to dispatch operations. (*See, e.g.,* FIG. 1, element 104; and p. 12, line 14.)

The microprocessor also includes a scheduler coupled to the dispatch unit and configured to schedule dispatched operations for execution. (*See, e.g.,* FIG. 1, element 118; and p. 12, lines 15-16.)

In response to receiving a microcoded instruction, the dispatch unit is configured to dispatch to the scheduler a microcode subroutine call operation that includes a tag identifying a microcode subroutine associated with the microcoded instruction. (*See, e.g.,* FIG. 13C; p. 41 lines 23-28, and p. 42, lines 6-9.)

Independent claim 29 is directed to a method including receiving a stream of instructions and detecting a microcoded instruction within the stream of instructions, wherein the microcoded instruction immediately precedes an other instruction in program order. (*See, e.g.,* FIGs. 13A and 14; p. 39, lines 22-24; and p. 45, lines 13-14.)

The method also includes, in response to detecting a microcoded instruction, dispatching a microcode subroutine call operation that identifies a microcode subroutine associated with the microcoded instruction. (*See, e.g.,* FIG. 13C; FIG. 14, element 1405; and p. 41, lines 23-28.) The microcode subroutine call operation pushes an address of the other instruction onto a stack. (*See, e.g.,* FIG. 14, element 1405; and p. 45, lines 27-29.)

The method further includes executing a plurality of operations included in the microcode subroutine, wherein the plurality of operations includes a return operation, and wherein execution of the return operation pops the address from the stack. (*See, e.g.,* FIG. 14, elements 1403 and 1407; p. 42, lines 11-13; and p. 46, lines 15-18.)

Independent claim 40 is directed to a method including dispatching one or more operations included in a first microcode subroutine and one or more operations included in a second microcode subroutine, wherein dispatching the one or more operations in the first microcode subroutine includes performing register name replacements using replacement register names stored in a first alias table element and wherein dispatching the one or more operations in the second microcode subroutine includes performing register name replacements using replacement register names stored in a second alias table element. (*See, e.g.,* p. 43, lines 7-19; and p. 45, lines 2-6.)

The method also includes, subsequent to dispatching the microcode subroutines, detecting a branch misprediction within the first microcode subroutine. (*See, e.g.,* p. 44 lines 10-12.) In response to detecting a branch misprediction, the method includes replacing register names within one or more other operations included in the first microcode subroutine with replacement register names stored in the first alias table element and dispatching the one or more other operations subsequent to replacing the register names. (*See, e.g.,* p. 44, lines 10-31.)

Independent claim 41 is directed to a system similar to that of claim 17. The system includes means for receiving a stream of instructions, decoding each non-microcoded instruction within the stream of instructions into one or more operations, and dispatching each of the one or more operations. (*See, e.g.,* FIG. 1, dispatch unit 104, which includes decode unit 140; FIG. 13A; p. 11, line 29 – p. 12, line 1; and p. 39, lines 22-24.)

The system includes means for executing dispatched operations. (*See, e.g.,* FIG. 1, dispatch unit 104, which includes microcode unit 150; FIG. 1, execution cores 124; p. 11, lines 4-5; p. 12, lines 15-18; and p. 13, lines 21-23.)

The means for receiving the stream of instructions are further configured to detect a microcoded instruction within the stream of instructions and to responsively dispatch a microcode subroutine call operation that identifies a microcode subroutine associated with the microcoded instruction. (*See, e.g.,* FIG. 1, dispatch unit 104, which includes microcode unit 150; p. 13, lines 12-14 and 21-22; and p. 42, lines 6-9.)

The means for executing dispatched operations are configured to push an address onto a stack when executing the microcode subroutine call operation, wherein the address identifies an operation generated by decoding a non-microcoded instruction immediately subsequent to the microcoded instruction within the stream of instructions. (*See, e.g.,* FIG. 1, microcode unit 150; FIG. 13A; and p. 41, lines 1-5.)

The summary above describes various examples and embodiments of the claimed subject matter; however, the claims are not necessarily limited to any of these examples and embodiments. The claims should be interpreted based on the wording of the respective claims.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 1, 15-17, 27-29, 39 and 41 stand finally rejected under 35 U.S.C. § 102(b) as being anticipated by Tran (U.S. Patent 5,864,689) (also referencing "Intel Architecture Software Developer's Manual (hereinafter "Manual").

2. Claim 40 stands finally rejected under 35 U.S.C. § 102(b) as being anticipated by Carbine et al. (U.S. Patent 5,630,083) (hereinafter "Carbine").

3. Claims 2-6, 18-22 and 30-34 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Tran in view of Carbine.

4. Claims 7, 8, 23, 24, 35 and 36 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Tran and Carbine and further in view of Rotenberg, et al. ("Trace Cache: a Low Latency Approach to High Bandwidth Instruction Fetching") (hereinafter "Rotenberg").

5. Claims 9, 10, 25 and 37 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Tran in view of Kling (U.S. Publication 2004/0049657).

6. Claims 11-14, 26 and 38 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Tran and Kling and further in view of Harris (U.S. Patent 6,260,138).

VII. ARGUMENT

First ground of rejection:

Claims 1, 15-17, 27-29, 39 and 41 stand finally rejected under 35 U.S.C. § 102(b) as being anticipated by Tran (U.S. Patent 5,864,689) (also referencing "Intel Architecture Software Developer's Manual (hereinafter "Manual"). Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 1, 15, 16, 17, 27, 28, and 39:

Regarding claim 1, contrary to the Examiner's assertion, Tran clearly fails to disclose *a scheduler coupled to the dispatch unit and configured to schedule dispatched operations for execution, wherein in response to receiving a microcoded instruction, the dispatch unit is configured to dispatch to the scheduler a microcode subroutine call operation that includes a tag identifying a microcode subroutine associated with the microcoded instruction*. The Examiner cites column 8, line 55 – column 9, line 4, and Tran's target address therein, as teaching that in response to receiving a microcoded instruction, the dispatch unit is configured to dispatch to the scheduler a microcode subroutine call operation that includes a tag identifying a microcode subroutine associated with the microcoded instruction. However, this citation does not describe Tran's microcode unit 45 and instruction decode unit 36 (which the Examiner equates with Appellants' dispatch unit) dispatching to a reservation station (which the Examiner equates with Appellants' scheduler) a microcode subroutine call operation including the limitations recited in claim 1. Instead, this citation describes, "when instruction decode unit 36 detects an instruction corresponding to a routine within microcode unit 45, an indication of the instruction is transferred upon instruction indication bus 62 to microcode unit 45" and also "the indication may comprise the target address of a subroutine call instruction." In Tran, the instruction that is detected is a source-code instruction, e.g., "CALL", having a target address in a particular range.

Therefore, Tran does not describe detecting a **microcoded** instruction, but detecting a non-microcoded instruction (e.g., a "CALL" instruction) that is used to invoke a subroutine call in order to emulate a microcoded instruction (e.g., to emulate an instruction that is not included in the instruction set.) In addition, the Examiner appears to equate Tran's "target address" with the "tag" of Appellants' claim 1. However, an indication of an instruction, even one that includes a target address of a subroutine call instruction (or tag), is not itself a microcode subroutine call operation, as recited in claim 1.

Furthermore, this citation describes that when instruction decode unit 36 detects a microcoded instruction, it sends an indication of the instruction to microcode unit 45. It does not describe dispatching a microcode subroutine call operation *to a scheduler*, as recited in claim 1. Note that according to Tran, the indication of the instruction is sent to microcode unit 45, not to Tran's reservation station (which the Examiner equates to the scheduler of claim 1). Tran's microcode unit 45 is not a scheduler. In fact, column 8, lines 61-64 goes on to describe that if instruction decode unit 36 detects an instruction to be performed by microcode unit 45, it stalls (i.e., stops dispatching instructions) until microcode unit 45 completes the corresponding routine.

In the Response to Arguments section of the Final Office Action, the Examiner submits:

> The execution of the x86 CALL instruction involves the storing of context information (such as the instruction pointer) in addition to the execution of the routine (microcode subroutine instructions in the context of Tran)... In order for context information to be stored, the x86 CALL instruction itself must be executed by one of the execution units of the processor. Execution by one of the execution units requires that the instruction be dispatched from decode unit (part of the dispatch unit) to the reservation station (i.e., the scheduler) associated with the desired execution unit.

Appellants note that the Examiner has cited no evidence that execution of the CALL instruction requires any of these specific actions by any of the specific components recited in Appellants' claims, as a CALL instruction may be implemented in a microprocessor using any of a wide variety of actions by various components of the microprocessor. In addition, the Examiner's remarks appear to acknowledge that the CALL instruction itself is not a <u>microcoded instruction</u> for which the operations recited in claim 1 are performed. For example, the Examiner states that the "CALL instruction itself must be <u>executed by one of the execution units</u>," and that it is dispatched to the scheduler associated with the desired execution unit. In Tran, microcoded instructions are not dispatched to one of the execution units 38 for execution, but are handled by microcode unit 45. This is clearly not the case for the execution of the CALL instruction itself, as noted in the Examiner's own remarks.

In addition, dispatching <u>the CALL instruction itself</u> to the scheduler of an execution unit 38, as the Examiner describes, is in direct contrast to the limitations of claim 1, in which *in response to receiving <u>a microcoded instruction</u>, the dispatch unit is configured to dispatch to the scheduler <u>a microcode subroutine call operation</u>*. This claim clearly indicates that the received microcoded instruction and the dispatched microcode subroutine call operation are not the same instruction/operation. In response to receiving one (the microcoded instruction), the other (the microcode subroutine call operation) is dispatched. By contrast, in Tran, the x86 CALL instruction, a source code instruction of the x86 instruction set architecture, is received and is itself dispatched for execution.

Furthermore, contrary to the Examiner's assertions, the x86 CALL instruction is not a <u>microcoded instruction</u>, as would be understood by anyone of ordinary skill in the art, and Tran does not teach that the CALL instruction itself is a microcoded instruction (i.e., one executed in microcode). Instead, the CALL instruction is described as an example of a <u>branch instruction</u>, which also cannot be considered a microcoded instruction (see, e.g., Tran claim 5, and column 4, lines 44 – 54, in which a standard subroutine call and return are described.) Tran teaches that if a branch type opcode (including a CALL opcode, for example) is detected and the target address of the branch is within a particular range of addresses, the target address may be routed to microcode unit 45. This is clearly not the same as a dispatch unit performing the functions recited in claim 1 (which are not taught by Tran) <u>in response to receiving a microcoded instruction</u>, as required by Appellants' claims. There is nothing in Tran that teaches a dispatch unit dispatching to the scheduler a microcode subroutine call operation in response to *receiving a microcoded instruction*.

Anticipation requires the presence in a single prior art reference disclosure of each and every limitation of the claimed invention, arranged as in the claim. M.P.E.P 2131; *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 221 USPQ 481, 485 (Fed. Cir. 1984). The **identical** invention must be shown in as complete detail as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). As discussed above, Tran clearly fails to disclose *in response to receiving a microcoded instruction, the dispatch unit is configured to dispatch to the scheduler a microcode subroutine call operation that includes a tag identifying a microcode subroutine associated with the microcoded instruction*. Therefore, Tran certainly cannot be said to anticipate claim 1.

For at least the reasons above, the rejection of claim 1 is not supported by the cited art and removal thereof is respectfully requested. Claim 17 includes limitations similar to claim 1, and so the arguments presented above apply with equal force to this claim, as well.

Claim 29 and 41:

Regarding claim 29, contrary to the Examiner's assertion, Tran clearly fails to disclose *detecting a microcoded instruction within the stream of instructions, wherein the microcoded instruction immediately precedes an other instruction in program order, and in response to said detecting, dispatching a microcode subroutine call operation that identifies a microcode subroutine associated with the microcoded instruction.* First, as discussed above, Tran does not teach detecting a microcoded instruction, but detecting a non-microcoded instruction that emulates a microcoded instruction. The Examiner cites column 4, lines 36-41, as teaching detecting a microcoded instruction that immediately precedes another instruction in program order.

However, this citation describes that if instruction decode unit 36 detects an instruction to be performed by microcode unit 45, it stalls (i.e., stops dispatching instructions) until microcode unit 45 indicates that the routine corresponding to that instruction has completed dispatch. This citation says nothing about the microcoded instruction immediately preceding another instruction in program order, as recited in Appellants' claim 29.

Further regarding claim 29, the Examiner cites column 8, line 55 – column 9, line 4, as teaching dispatching a microcode subroutine call operation in response to detecting the microcoded instruction. However, this citation does not describe dispatching a microcode subroutine call operation. In fact, as described above, Tran **teaches away from** the dispatch unit dispatching a microcode subroutine call operation in response to detecting a microcoded instruction in the previous citation (instruction decode unit 36 stops dispatching instructions). Instead, this citation describes instruction decode unit 36 sending an indication of the detected instruction to microcode unit 45, which completes the routine. As discussed above regarding claim 1, sending an indication of a detected instruction is clearly not the same as dispatching a microcode subroutine call operation, as recited in Appellants' claim 29.

Finally regarding claim 29, the Examiner cites column 4, lines 42-54 as teaching that the microcode subroutine call operation pushes an address of the other instruction onto a stack, that the microcode subroutine includes a return operation, and that execution of the return operation pops the address from the stack (x86 Call instruction and x86 Return instruction). However, this citation has nothing to do with pushing or popping an instruction onto or off of a stack as part of a microcode subroutine call operation dispatched in response to detecting a microcoded instruction, as recited in claim 29. Instead it describes that source-code level subroutine call instructions (e.g., the CALL instruction of the x86 instruction set) having target addresses within a particular range of addresses are indicative of DSP functions. In this example, these DSP functions are executed by branching to a subroutine explicitly programmed using the CALL and RET instructions of the x86 instruction set. That is, the instruction detected by instruction decode unit 36 is itself a CALL instruction to a particular target address range, not a microcoded instruction for which a microcode subroutine call operation is dispatched in response to this detection. As discussed above, the x86 CALL instruction is not a microcoded instruction, but is an example of a branch-type instruction, which is detected in Tran.

In the Response to Arguments section of the Final Office Action, the Examiner argues that Tran has taught that a subroutine call instruction is dispatched in response to detecting a microcoded instruction. As discussed above, this is incorrect. The Examiner's citation in column 4, lines 47-52, describes a standard (source-code level) subroutine call that emulates a microcoded instruction, not a microcode subroutine call and return of a microcoded instruction. In addition, in Tran, the dispatch unit stops fetching and dispatching operations while microcode unit 45 completes the microcode routine and then resumes fetching and dispatching operations in response to assertion of the signal complete line 64. Nothing in Tran discloses the microcode subroutine pushing and popping operations on a stack.

The Examiner further argues that, "the CALL instruction is a microcoded instruction as the operation to be executed in response to the instruction is, at least partially, contained within microcode... That is, the CALL instruction invokes microcode" and cites Tran, column 4, lines 36-54. First, the Examiner has provided no evidence for this definition of a microcoded instruction, nor is there any basis for this definition found in the cited art. Furthermore, the Examiner's characterization of these passages in Tran is incorrect. Column 4, lines 36 – 41 merely describes that instruction decode unit 36 stalls upon detection of an instruction to be performed by microcode unit 45, and column 4, lines 42 – 54 describe a standard (source-code level) subroutine call and return that may be used to emulate a microcoded instruction, not a microcoded instruction itself. Neither of these passages discloses that a CALL instruction can itself be considered a microcoded instruction as would be understood by one of ordinary skill in the art. As discussed above, a CALL instruction is an example of a branch-type instruction and is not a microcoded instruction, as required by Appellants' claim.

Finally, the Examiner submits that the x86 CALL instruction is <u>both</u> a microcoded instruction and a microcode subroutine call operation. This is clearly illogical and is not taught by Tran. In Tran, the CALL instruction is not a <u>microcode subroutine call operation</u>, nor is it dispatched to a scheduler <u>in response to detection of a microcoded instruction</u>. Instead, Tran teaches that a target address of a CALL instruction (which is also clearly not a microcode subroutine call <u>operation</u>) is sent to microcode unit 45 in response to detection of <u>a branch-type instruction</u> (the CALL instruction itself). Appellants' claims clearly indicate that the received and/or detected microcoded instruction and the dispatched microcode subroutine call operation are not the same instruction/operation. In response to receiving and/or detecting one (the microcoded instruction), the other (the microcode subroutine call operation) is dispatched. By contrast, in Tran, the x86 CALL instruction, a source code instruction of the x86 instruction set architecture, is received and is itself dispatched for execution.

For at least the reasons above, the rejection of claim 29 is not supported by the cited art and removal thereof is respectfully requested. Claim 41 includes limitations similar to claim 29, and so the arguments presented above apply with equal force to this claim, as well.

<u>**Second ground of rejection:**</u>

Claim 40 stands finally rejected under 35 U.S.C. § 102(b) as being anticipated by Carbine et al. (U.S. Patent 5,630,083) (hereinafter "Carbine"). Appellants traverse this rejection for at least the following reasons.

Carbine clearly fails to disclose *dispatching one or more operations included in a first microcode subroutine and one or more operations included in a second microcode subroutine, wherein said dispatching the one or more operations in the first microcode subroutine comprises performing register name replacements using replacement register names stored in a first alias table element and wherein said*

dispatching the one or more operations in the second microcode subroutine comprises performing register name replacements using replacement register names stored in a second alias table element. The Examiner cites column 12, lines 35-56, as teaching generic microcode routines, and asserts that each of these generic microcode routines "has micro-alias registers" to replace register names within the routine. The Examiner seems to be implying that these generic routines, including what he interprets as separate micro-alias registers, teach <u>dispatching operations from both a first and second microcode subroutine</u>, and the use of both <u>a first and second alias table element</u>. However, there is nothing in this citation or elsewhere in Carbine that teaches <u>multiple alias table elements</u> or multiple micro-alias registers. Instead, Carbine describes a <u>single micro-alias register</u> (having multiple fields, but not multiple entries), which is <u>loaded with different data</u> dependent on which of four CUOPs is selected by multiplexer 560. There is also nothing in this citation that teaches <u>dispatching multiple microcode subroutines</u>, as recited in claim 40. Instead, Carbine describes microcode sequencing unit 534 issuing multiple CUOPs for one microcode flow (associated with a single entry point) at a time (see, e.g., column 11, lines 11-13). Since Carbine fails to teach or suggest dispatching operations from a first and second microcode subroutine and a first and second alias table element, Carbine cannot be said to anticipate claim 40.

In the Response to Arguments section of the Final Office Action, the Examiner submits that Carbine specifically refers to a plurality of micro-alias registers at column 12, lines 36-39. The Examiner also submits that Carbine states that registers are not hard-coded into any routine, "thereby indicating multiple generic microcode routines are used. Therefore, Carbine has taught dispatching multiple generic microcode routines." First, Appellants assert that the Examiner has misinterpreted Carbine's reference to "micro-alias registers." The Examiner's citation at column 12, lines 36-39 states, "The micro-alias registers 562 are particularly useful in long instruction flows, which allows the microcode programmer flexibility in retaining information and simplifying code." As is apparent from reading the rest of this passage, Carbine is referring to the separate storage locations (i.e., the SRC1, SRC2, and DEST fields) within the micro-alias register, not to multiple sets of such register fields in a plurality of micro-alias registers. Carbine refers to "the micro-alias register 562" and "the micro-alias registers 562" interchangeably throughout the specification. Carbine does not ever refer to multiple sets of such fields contained in multiple instances of micro-alias register 562, much less multiple instances of micro-alias register 562 each of which is associated with a different microcode routine, as the Examiner suggests.

In addition, Appellants assert that even if multiple generic routines may exist in the system of Carbine, there is nothing in Carbine that teaches <u>dispatching</u> two such routines. While the system of Carbine is directed toward parallel decoding of multiple instructions (see Abstract) there is nothing in Carbine that discloses <u>dispatching operations from two different microcode subroutines</u>, as required in Claim 40. Appellants assert that it is clearly not necessary that a system support <u>dispatching</u> operations from two different routines even if they may be decoded in parallel, and the Examiner has not cited anything in Carbine that discloses this limitation of Appellants' claim.

For at least the reasons above, the rejection of claim 40 is not supported by the cited art and removal thereof is respectfully requested.

<u>Third ground of rejection:</u>

Claims 2-6, 18-22 and 30-34 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Tran in view of Carbine. Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

<u>Claims 2, 18, and 30:</u>

Regarding claim 2, contrary to the Examiner's assertion, Tran in view of Carbine clearly fails to teach or suggest *wherein the dispatch unit is further configured to dispatch an operation that provides one or more register names for use as replacement register* *names within the microcode subroutine.* The Examiner admits that Tran does not explicitly teach this limitation and relies on Carbine to teach it. The Examiner submits that Carbine has taught a dispatch unit (microcode sequencing unit; FIG. 5, component 534) that is configured to dispatch an operation (LOADUAR signal) that provides one or more register names for use as replacement register names within the microcode subroutine (Carbine, column 12, lines 24-67).

First, Appellants note that the microcode sequencing unit of Carbine is clearly not analogous to the <u>dispatch unit</u> of Appellants' claims, as it does not perform the functions recited in claim 1. For example, the microcode sequencing unit is not *configured to dispatch operations*, much less to *dispatch to the scheduler a microcode subroutine call operation that includes a tag identifying a microcode subroutine associated with the microcoded instruction*. In fact, Carbine is directed to an instruction decoder and there is no description in Carbine about how, when, or where decoded instructions are dispatched for execution.

In addition, Appellants assert that the LOADUAR ("Load Micro-Alias Register") signal is not an <u>operation dispatched by a dispatch unit to a scheduler</u> (i.e., an operation dispatched for execution). Instead, it is a <u>signal</u> supplied by the microcode sequencing unit 534 to load the micro-alias register from one of four Cuops (as selected by a UARUIP signal). Furthermore, this signal clearly does not <u>provide register names</u> for replacement register names within a microcode subroutine, and no such replacement is described in Carbine. Instead, Carbine describes that microcode subroutines may be written such that they may be shared by multiple microcode routines by <u>coding them</u> to be generic. In the example described in the Examiner's citation, <u>a microcode routine</u> stores the address of a register in which a number to be rounded is stored in the micro-alias registers and then a rounding subroutine is run, using the contents of the micro-alias register to locate the number to be rounded. This has absolutely nothing to do with Appellants' claim, in which ***<u>an operation</u>*** *that <u>provides one or more register names</u> for use as replacement register names within the microcode subroutine* is <u>dispatched by a dispatch unit to a scheduler for execution</u>.

The Examiner submits that at the time the invention was made, it would have been obvious to a person of ordinary skill in the art to modify the dispatch unit of Tran to dispatch an operation that provides one or more register names for use as replacement register names with the microcode subroutine as taught by Carbine and that the suggestion/motivation for doing so would have been that a generic microcode routine can be used by any number of other microcode programs (Carbine, column 12, lines 49-52). Appellants assert, however, that Carbine does not teach this limitation, as discussed above. Therefore, modifying Tran to include the LOADUAR signal, or allowing generic microcode routines to be shared, would not result in Appellants' claimed invention.

To establish a *prima facie* obviousness of a claimed invention, all claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 U.S.P.Q. 580 (C.C.P.A. 1974), MPEP 2143.03. Obviousness cannot be established by combining or modifying the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion or incentive to do so. *In re Bond,* 910 F. 2d 81, 834, 15 USPQ2d 1566, 1568 (Fed. Cir. 1990).

For at least the reasons above, the rejection of claim 2 is not supported by the cited art and removal thereof is respectfully requested. Claims 18 and 30 include limitations similar to claim 2, and so the arguments presented above apply with equal force to these claims, as well.

Claims 3, 19, and 31:

Regarding claim 3, contrary to the Examiner's assertion, Tran in view of Carbine clearly fails to teach or suggest *wherein the dispatch unit is configured to allocate an alias table element to store the one or more register names in response to handling the operation*. The Examiner cites Carbine (micro-alias register; FIG. 5, component 562; and column 12, lines 15-35) as teaching this limitation. The Examiner's cited passage describes that the micro-alias register 562 is loaded with three values (in fields SRC1, SRC2, and DEST) from one or more Cuops, dependent on a 4:1 multiplexer 560. The micro-alias register is clearly not a <u>table in which elements are allocated</u>, much less one that is configured to <u>allocate an element</u> to store register names <u>in response to handling an operation that provides one or more register names</u> for use as replacement register names within the microcode subroutine and that is dispatched by a dispatch unit to a scheduler for execution. In addition, there is nothing in Carbine that teaches or suggests that a dispatch unit (i.e., one that performs the functions recited in Appellants' claims) <u>allocates</u> a micro-alias register for any reason. Instead, the micro-alias register is always available to be loaded with information from one of four Cuop registers 550. Therefore, Appellants assert that Tran and Carbine, taken alone or in combination, do not teach or suggest this limitation of claim 3.

In addition, the Examiner has failed to provide a reason why a person of ordinary skill in the art would be motivated to combine the teachings of Tran and Carbine in teaching the limitations of claim 3. The broad statements made by the Examiner regarding claim 2 do not include a suggestion in the cited prior art for such a specific combination and include no mention of the specific limitations of claim 3.

For at least the reasons above, the rejection of claim 3 is not supported by the cited art and removal thereof is respectfully requested. Claims 19 and 31 include limitations similar to claim 3, and so the arguments presented above apply with equal force to these claims, as well.

<u>Claims 4-5, 20-21, and 33-34:</u>

Contrary to the Examiner's assertion, Tran in view of Carbine clearly fails to teach or suggest *wherein the dispatch unit is configured to maintain multiple allocated alias table elements at a same time* (as recited in claim 4) or *wherein each of the multiple allocated alias table elements is associated with a respective microcode subroutine, wherein the dispatch unit is configured to maintain each alias table element at least until all branch operations within the respective microcode subroutine have resolved* (as recited in claim 5.) The Examiner again cites the micro-alias register 562 of Carbine and column 12, lines 15-35 as teaching the alias table of these claims. However, as discussed above, the micro-alias register of Carbine is clearly not an element of <u>an alias table</u>, which is <u>allocated</u> in response to <u>an operation that provides register names</u> for replacement register names within a microcode routine, as required by Appellants' claims. In addition, as discussed above regarding claim 40, Carbine does not teach or suggest <u>multiple instances</u> of micro-alias register 562, as was suggested by the Examiner. Therefore, Carbine clearly cannot, and does not, teach or suggest <u>maintaining multiple allocated alias table elements at a same time</u>, as recited in claim 4.

In addition, the Examiner cites Carbine (column 23, lines 12-35) as teaching "each of the multiple allocated alias table elements (micro-alias registers) is associated with a respective microcode subroutine (Carbine; The generic microcode routine that uses that particular allocated micro-alias register; See column 12, lines 49-56)". The Examiner has misinterpreted this passage of Carbine, which is reproduced below:

> "For example, the microcode may include a rounding routine that is generically used in many different microcode sequences to round a number. Because the number to be rounded will likely be stored by a microcode instruction not within the rounding routine, the rounding sequence itself would not know directly where the information was stored. Therefore, <u>the preceding microcode stores the address of this number</u> (i.e., the register in which it is stored) <u>in the micro-alias registers 562</u>, and then <u>the rounding routine is run, using the contents of the micro-alias register 562 to specify the register address of a number to be rounded</u>. Thus, by storing the address of the register in which the data is stored, a common generic microcode routine can be utilized by any of a number of other microcode programs. Because the registers are not hard-coded into any routine, the programmer has great flexibility to access any register indirectly by accessing the register address within the micro-alias register 562."

This passage does not teach that each of multiple allocated alias table elements (micro-alias registers) is associated with a respective microcode subroutine, as the Examiner suggests. Instead, it describes that a microcode routine that will call a generic rounding subroutine stores a number to be rounded in a register and stores an indication of where the number to be rounded is stored in the micro-alias register (e.g., as a value for the SRC1 field) before calling the generic rounding subroutine. The generic rounding subroutine is coded to examine the SRC1 field in order to locate the number to be rounded, in this example. In this example, other microcode routines may share this rounding subroutine and would also store in the SRC1 field an indication of where a number to be rounded is stored. This clearly does not teach or suggest anything about each of multiple allocated alias table elements being associated with a respective microcode subroutine.

Finally, the Examiner submits that Carbine teaches the dispatch unit is configured to maintain each alias table element at least until all branch operations within the respective microcode subroutine have resolved, "If a micro-branch (a branch in microcode) is mispredicted the dispatch unit updates/maintains the micro-alias registers so that execution can restart at the actual target of the micro-branch. Therefore, the dispatch unit maintains each micro-alias register until all branch operations within the respective microcode subroutine have resolved; See column 23, lines 12-35". **Appellants assert that the Examiner's citation, as well as his own remarks, teaches away from these limitations of Appellants' claims**. First, the Examiner's remarks include the phrase, "the dispatch unit updates/maintains the micro-alias registers." Appellant asserts that updating the micro-alias registers is the **opposite** of maintaining alias table elements. The term "maintain" is used in this context to mean that the value is not changed, as is clear in both the recitation of Appellants' claims and in the accompanying description in Appellants' specification.

Similarly, the Examiner's citation in column 23 states, in part, that in response to a mispredicted branch, "If either macro-alias data or micro-alias data or both may be used by subsequent micro-operations, then the first instructions at the target microcode flow restore the state by restoring the macro-alias registers if necessary and by restoring the micro-alias registers if necessary for use by subsequent micro-operations." This clearly indicates that in Carbine, the micro-alias registers (i.e., the fields of micro-alias register 562) must be restored because the values were not maintained until the branch was resolved. Therefore, Carbine teaches away from the above-referenced limitation of claim 5.

Appellants assert, therefore, that Tran and Carbine, taken alone or in combination, do not teach or suggest this limitation of claims 4 and 5.

In addition, the Examiner has failed to provide a reason why a person of ordinary skill in the art would be motivated to combine the teachings of Tran and Carbine in teaching the limitations of claims 4 and 5. The broad statements made by the Examiner regarding claim 2 do not include a suggestion in the cited prior art for such a specific combination and include no mention of the specific limitations of claims 4 and 5.

For at least the reasons above, the rejection of claims 4 and 5 is not supported by the cited art and removal thereof is respectfully requested. Claims 20-21 and 33-34 include limitations similar to those of claims 4 and 5, and so the arguments presented above apply with equal force to these claims, as well.

<u>Claims 6, 22, and 32:</u>

Regarding claim 6, contrary to the Examiner's assertion, Tran in view of Carbine clearly fails to teach or suggest *wherein in response to detection of a branch misprediction within a microcode subroutine, the dispatch unit is configured to perform replacements within one or more microcode operations included within the microcode subroutine according to the one or more register names stored within a respective alias table element and to dispatch the one or more microcode operations subsequent to performing the replacements*. The Examiner again cites Carbine (column 23, lines 12-35) as teaching these limitations. However, as discussed above, Tran in view of Carbine does not teach or suggest the dispatch unit and alias table elements of Appellants' invention, or the register names stored in the alias table elements. In addition, this passage of Carbine describes that in order to restore the macro-alias and micro-alias registers after a branch misprediction, the macroinstruction must first be <u>re-fetched</u> and

<u>re-decoded</u> (as illustrated in FIG. 12) and then the first instructions at the target microcode routine may restore the macro-alias and micro-alias registers, if they will be needed by subsequent micro-operations. This clearly **teaches away from** the limitations of claim 6, in which the <u>one or more register names already stored within a respective alias table element</u> are used to perform replacements in response to detection of a branch misprediction.

For at least the reasons above, the rejection of claim 6 is not supported by the cited art and removal thereof is respectfully requested. Claims 22 and 32 include limitations similar to claim 6, and so the arguments presented above apply with equal force to these claims, as well.

<u>Fourth ground of rejection:</u>

Claims 7, 8, 23, 24, 35 and 36 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Tran and Carbine and further in view of Rotenberg, et al. ("Trace Cache: a Low Latency Approach to High Bandwidth Instruction Fetching") (hereinafter "Rotenberg"). Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

<u>Claims 7, 23, and 35:</u>

Regarding claim 7, contrary to the Examiner's assertion, Tran in view of Carbine and Rotenberg clearly fails to teach or suggest *comprising a trace cache coupled to the dispatch unit, wherein the trace cache includes a trace cache entry; wherein a trace stored in the trace cache entry includes the microcode subroutine call operation and the one or more register names for use as replacement register names*. Appellants assert that, for at least the reasons presented above, the cited references do not teach the microprocessor of claim 2 wherein the dynamic instruction stream includes a microcode subroutine call operation and the one or more register names for use as replacement register names, as the Examiner suggests.

Further regarding claim 7, the Examiner admits, "Tran and Carbine have not explicitly taught a trace cache coupled to the dispatch unit, wherein the trace cache includes a trace cache entry; wherein a trace stored in the trace cache entry includes instructions from the dynamic instruction stream". The Examiner submits that Rotenberg has taught a trace cache coupled to the dispatch unit, wherein the trace cache includes a trace cache entry and that a trace stored in the trace cache entry includes instructions from the dynamic stream. Appellants note that the Examiner has misquoted Appellants' claim. Claim 7 does not recite that a trace cache entry includes "instructions from the dynamic stream," but a trace cache entry that includes <u>the microcode subroutine call operation and the one or more register names for use as replacement register names.</u> There is nothing in Rotenberg or the other cited references that teaches a trace entry including these specific elements. Therefore Tran in view of Carbine and Rotenberg does not teach or suggest all the limitations of claim 7.

The Examiner submits that it would have been obvious to a person of ordinary skill in the art to modify the system of Tran and Carbine to include a trace cache coupled to the dispatch unit, wherein the trace cache includes a trace cache entry; wherein a trace stored in the trace cache entry includes instructions from the dynamic instruction stream as taught by Rotenberg because the trace cache would improve the performance of the microprocessor (Rotenberg, Abstract.) Appellants assert, however, that including the trace cache of Rotenberg in the system of Tran and Carbine would not teach or suggest all the limitations of claim 7, since Rotenberg's trace cache entries do not store the elements recited in claim 7.

For at least the reasons above, the rejection of claim 7 is not supported by the cited art and removal thereof is respectfully requested. Claims 23 and 35 include limitations similar to claim 7, and so the arguments presented above apply with equal force to these claims, as well.

Claims 8, 24, and 36:

Regarding claim 8, contrary to the Examiner's assertion, Tran in view of Carbine and Rotenberg clearly fails to teach or suggest *wherein in response to receiving the trace from the trace cache, the dispatch unit is configured to allocate an alias table to store the one or more register names*. The Examiner again cites Carbine (micro-alias register; FIG. 5, component 562; and column 12, lines 15-35) as teaching this limitation, "In response to receiving the LOADUAR instruction, a micro-alias register is allocated." However, as discussed above, LOADUAR is not an instruction at all (i.e., an operation dispatch to the scheduler for execution). Instead, it is a signal that causes micro-alias register 562 to be loaded with values from one of four Cuops. Therefore, the cited references clearly do not teach or suggest this limitation.

For at least the reasons above, the rejection of claim 8 is not supported by the cited art and removal thereof is respectfully requested. Claims 24 and 36 include limitations similar to claim 8, and so the arguments presented above apply with equal force to these claims, as well.

Fifth ground of rejection:

Claims 9, 10, 25 and 37 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Tran in view of Kling (U.S. Publication 2004/0049657). Appellants traverse this rejection for at least the following reasons.

Regarding claim 9, contrary to the Examiner's assertion, Tran in view of Kling clearly fails to teach or suggest *wherein the microcode subroutine is stored as one or more microcode traces*. The Examiner admits that Tran has not explicitly taught that the dispatch unit is configured to store the microcode subroutine in one or more microcode traces, and relies on Kling to teach this limitation. The Examiner submits that Kling has taught a dispatch unit (Kling; microcode unit; FIG. 2, component 46; paragraph 13) that is configured to store a microcode subroutine in one or more microcode traces (Kling; paragraph 32). First, component 46 of FIG. 2 does not illustrate a microcode unit, as the Examiner suggests, but microcode. Therefore, this component cannot be equated to the dispatch unit of Appellants' claims. In addition, the Examiner has mischaracterized the teachings of paragraph [0032] of Kling, which is reproduced below:

> [0032] Further optimization of the use of the extended register space 38 can be achieved with processors having trace cache-based microarchitectures. In particular, when a processor having a trace cache-based microarchitecture identifies an instruction that requires access to the extended register space 38, <u>information relating to that instruction and the extended register space to which it requires access can be stored in the microcode trace</u> to enable more efficient processing of that instruction during subsequent invocations of the instruction.

This passage mentions the existence of a "microcode trace" but does not teach or suggest that <u>**a microcode subroutine** is stored in one or more such traces</u>. The only other mention of a "trace" in Kling is found in claim 25, which includes the limitation, "using a microcode trace to store information associated with accessing the register space." Appellants assert that neither of these passages teaches or suggests the limitations of Appellants' claim 9.

For at least the reasons above, the rejection of claim 9 is not supported by the cited art and removal thereof is respectfully requested. Claims 25 and 37 include limitations similar to claim 9, and so the arguments presented above apply with equal force to these claims, as well.

<u>Sixth ground of rejection:</u>

Claims 11-14, 26 and 38 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Tran and Kling and further in view of Harris (U.S. Patent 6,260,138). Appellants traverse this rejection for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 11, 26, and 38:

Regarding claim 11, contrary to the Examiner's assertion, Tran in view of Kling and Harris clearly fails to teach or suggest *wherein each microcode operation stored in the one or more microcode traces includes an associated liveness indication*. The Examiner admits that Tran and Kling have not explicitly taught that each operation includes an associated liveness indication and relies on Harris to teach this limitation.

The Examiner submits that Harris has taught an instruction cache wherein each operation includes an associated liveness indication (Harris; priority tag; column 3, lines 28-31). Appellants assert, however, that the priority tag of Harris is not an indication of liveness, as this term would be understood by one of ordinary skill in the art or as used in Appellants' specification. As described in Appellants' specification, the liveness indication associated with each microcode operation identifies the branch operations within the microcode trace upon which that operation is dependent, i.e., it identifies the <u>liveness group</u> to which the operation belongs, dependent on whether the operation follows an unconditional or a particular one of one or more conditional branches within a trace. *(See, e.g.,* FIG. 2B; p. 5, lines 12-13; and p. 20, lines 4-22.)

The priority tags of Harris are assigned individually to each instruction and are used to determine an order in which to execute instructions on different branch paths. They may be dependent on the position of each instruction with respect to a predicted path (see, e.g., FIG. 7 of Harris). However, there is nothing in Harris that describes storing such an indication along with microcode operations in <u>a microcode trace</u>, as recited in Appellants' claims. Instead, Harris describes that the priority tags may be stored in <u>an instruction cache</u> (which is not a trace cache) or in a separate buffer, as in the Examiner's citation in Harris:

> In a preferred embodiment of the invention, a pre-decode unit is operable to determine instruction path priorities and to associate a priority tag with each instruction <u>in an instruction cache</u>. A decode unit can be operable to decode instructions from the instruction cache for transfer to the instruction buffer.

Harris does not describe a trace cache at all, much less a trace cache storing microcode subroutines. Furthermore, the Examiner has admitted that Tran has not explicitly taught that the dispatch unit is configured to store the microcode subroutine in one or more microcode traces (as discussed above regarding claim 9. Therefore, it is not clear how, or even if, the priority tags of Harris could be applied to Tran, or that the addition of priority tags would improve the performance of a trace cache including microcode subroutines (which is not found in Tran). Appellants assert that since Tran does not teach that microcode subroutines are stored in microcode traces, the combination of Tran and Harris cannot teach or suggest additional limitations on such microcode traces.

For at least the reasons above, the rejection of claim 11 is not supported by the cited art and removal thereof is respectfully requested. Claims 26 and 38 include limitations similar to claim 11, and so the arguments presented above apply with equal force to these claims, as well.

<u>**Claims 12, 13, and 14:**</u>

For at least the reasons presented above regarding claim 11, Tran in view of Kling and Harris also clearly fails to teach or suggest the limitations of claims 12, 13, and 14, which recite additional limitations on the use of <u>liveness indications stored with each microcode operation in one or more microcode traces</u>. Since it has been shown (above) that the cited references do not teach liveness indications stored with each microcode operation in one or more microcode traces, they clearly cannot, and do not, teach or suggest the additional limitations of claims 12, 13, and 14.

Appellants assert, therefore, that the rejection of claims 12, 13, and 14 is not supported by the cited art and removal thereof is respectfully requested.

CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-41 was erroneous, and reversal of his decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee of $500.00 and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5500-81600/RCK.

Respectfully submitted,

/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
Attorney for Appellants

Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
(512) 853-8850

Date: December 17, 2006

VIII. CLAIMS APPENDIX

The claims on appeal are as follows.

1. A microprocessor, comprising:

a dispatch unit configured to dispatch operations;

a scheduler coupled to the dispatch unit and configured to schedule dispatched operations for execution;

wherein in response to receiving a microcoded instruction, the dispatch unit is configured to dispatch to the scheduler a microcode subroutine call operation that includes a tag identifying a microcode subroutine associated with the microcoded instruction.

2. The microprocessor of claim 1, wherein the dispatch unit is further configured to dispatch an operation that provides one or more register names for use as replacement register names within the microcode subroutine.

3. The microprocessor of claim 2, wherein the dispatch unit is configured to allocate an alias table element to store the one or more register names in response to handling the operation.

4. The microprocessor of claim 2, wherein the dispatch unit is configured to maintain multiple allocated alias table elements at a same time.

5. The microprocessor of claim 4, wherein each of the multiple allocated alias table elements is associated with a respective microcode subroutine, wherein the dispatch unit is configured to maintain each alias table element at least until all branch operations within the respective microcode subroutine have resolved.

6. The microprocessor of claim 4, wherein in response to detection of a branch misprediction within a microcode subroutine, the dispatch unit is configured to perform replacements within one or more microcode operations included within the microcode subroutine according to the one or more register names stored within a respective alias table element and to dispatch the one or more microcode operations subsequent to performing the replacements.

7. The microprocessor of claim 2, further comprising a trace cache coupled to the dispatch unit, wherein the trace cache includes a trace cache entry; wherein a trace stored in the trace cache entry includes the microcode subroutine call operation and the one or more register names for use as replacement register names.

8. The microprocessor of claim 7, wherein in response to receiving the trace from the trace cache, the dispatch unit is configured to allocate an alias table to store the one or more register names.

9. The microprocessor of claim 1, wherein the microcode subroutine is stored as one or more microcode traces.

10. The microprocessor of claim 9, wherein the one or more microcode traces are stored within a read only memory.

11. The microprocessor of claim 9, wherein each microcode operation stored in the one or more microcode traces includes an associated liveness indication.

12. The microprocessor of claim 11, wherein the dispatch unit is configured to determine whether each microcode operation stored in one of the one or more microcode traces is executable dependent on at least one of: a branch prediction and the associated liveness indication;

wherein the dispatch unit is configured to signal whether each microcode operation stored in the one of the one or more microcode traces is executable when dispatching that microcode operation to the scheduler;

wherein the scheduler is configured to store an associated indication for each dispatched microcode operation indicating whether that dispatched microcode operation is executable.

13. The microprocessor of claim 12, wherein if the branch prediction is incorrect, the scheduler is configured to update the associated indication for at least one dispatched microcode operation.

14. The microprocessor of claim 11, wherein the dispatch unit is configured to selectively dispatch microcode operations included in the one or more microcode traces dependent upon at least one of: the associated liveness indication and a branch prediction.

15. The microprocessor of claim 1, wherein a same opcode is used to specify the microcode subroutine call operation and a non-microcode subroutine call operation.

16. The microprocessor of claim 1, wherein the microcode subroutine includes a return operation, wherein the return operation pops a return address from a stack, wherein execution of the microcode subroutine call operation pushes the return address onto the stack.

17. A computer system, comprising:

a system memory; and

a microprocessor coupled to the system memory, wherein the microprocessor comprises:

a dispatch unit configured to dispatch operations;

a scheduler coupled to the dispatch unit and configured to schedule dispatched operations for execution;

wherein in response to receiving a microcoded instruction, the dispatch unit is configured to dispatch to the scheduler a microcode subroutine call operation that includes a tag identifying a microcode subroutine associated with the microcoded instruction.

18. The computer system of claim 17, wherein the dispatch unit is further configured to dispatch an operation that provides one or more register names for use as replacement register names within the microcode subroutine.

19. The computer system of claim 18, wherein the dispatch unit is configured to allocate an alias table element to store the one or more register names in response to handling the operation.

20. The computer system of claim 18, wherein the dispatch unit is configured to maintain multiple allocated alias table elements at a same time.

21. The computer system of claim 20, wherein each of the multiple allocated alias table elements associated with a respective microcode subroutine, wherein the dispatch unit is configured to maintain each alias table element at least until all branch operations within the respective microcode subroutine have resolved.

22. The computer system of claim 20, wherein in response to detection of a branch misprediction within a microcode subroutine, the dispatch unit is configured to perform replacements within one or more microcode operations included within the microcode subroutine according to the one or more register names stored within a

respective alias table element and to dispatch the one or more microcode operations subsequent to performing the replacements.

23. The computer system of claim 18, further comprising a trace cache coupled to the dispatch unit, wherein the trace cache includes a trace cache entry; wherein a trace stored in the trace cache entry includes the microcode subroutine call operation and the one or more register names for use as replacement values.

24. The computer system of claim 23, wherein in response to receiving the trace from the trace cache, the dispatch unit is configured to allocate an alias table to store the one or more register names.

25. The computer system of claim 17, wherein the dispatch unit is configured to store the microcode subroutine in one or more microcode traces.

26. The computer system of claim 25, wherein each microcode operation stored in the one or more microcode traces includes an associated liveness indication.

27. The computer system of claim 17, wherein a same opcode is used to specify the microcode subroutine call operation and a non-microcode subroutine call operation.

28. The computer system of claim 17, wherein the microcode subroutine includes a return operation, wherein the return operation pops a return address from a stack, wherein execution of the microcode subroutine call operation pushes the return address onto the stack.

29. A method, comprising:

receiving a stream of instructions;

detecting a microcoded instruction within the stream of instructions, wherein the microcoded instruction immediately precedes an other instruction in program order;

in response to said detecting, dispatching a microcode subroutine call operation that identifies a microcode subroutine associated with the microcoded instruction, wherein the microcode subroutine call operation pushes an address of the other instruction onto a stack; and

executing a plurality of operations included in the microcode subroutine, wherein the plurality of operations include a return operation, and wherein execution of the return operation pops the address from the stack.

30. The method of claim 29, further comprising dispatching an operation that provides one or more register names for use as replacement register names within the microcode subroutine in response to said detecting.

31. The method of claim 30, further comprising allocating an alias table element to store the one or more register names in response to handling the operation that provides one or more register names for use as replacement register names.

32. The method of claim 31, further comprising replacing one or more register names within one or more microcode operations included in the microcode subroutine with the one or more register names from the alias table element in response to detection of a branch misprediction within the microcode subroutine.

33. The method of claim 30, further comprising maintaining multiple allocated alias table elements at a same time, wherein each of the multiple allocated alias table elements is associated with a different microcode subroutine.

34. The method of claim 33, wherein said maintaining comprises maintaining each alias table element at least until resolution of all branch operations within a respective microcode subroutine.

35. The method of claim 30, further comprising storing the microcode subroutine call operation and the one or more register names for use as replacement register names within a trace.

36. The method of claim 35, further comprising allocating an alias table element to store the one or more register names in response to fetching the trace from the trace cache.

37. The method of claim 29, further comprising storing the microcode subroutine in one or more microcode traces.

38. The method of claim 37, further comprising storing a liveness indication for each microcode operation stored in the one or more microcode traces.

39. The method of claim 29, further comprising dispatching a non-microcode subroutine call operation, wherein a same opcode is used to specify the microcode subroutine call operation and the non-microcode subroutine call operation.

40. A method, comprising:

dispatching one or more operations included in a first microcode subroutine and one or more operations included in a second microcode subroutine, wherein said dispatching the one or more operations in the first microcode subroutine comprises performing register name replacements using replacement register names stored in a first alias table element and wherein said dispatching the one or more operations in the second microcode subroutine comprises performing register name replacements using replacement register names stored in a second alias table element;

subsequent to said dispatching, detecting a branch misprediction within the first microcode subroutine;

in response to said detecting, replacing register names within one or more other operations included in the first microcode subroutine with replacement register names stored in the first alias table element; and

dispatching the one or more other operations subsequent to said replacing.

41. A system, comprising:

means for receiving a stream of instructions, decoding each non-microcoded instruction within the stream of instructions into one or more operations, and dispatching each of the one or more operations;

means for executing dispatched operations;

wherein the means for receiving the stream of instructions are configured to detect a microcoded instruction within the stream of instructions and to responsively dispatch a microcode subroutine call operation that identifies a microcode subroutine associated with the microcoded instruction;

wherein the means for executing dispatched operations are configured to push an address onto a stack when executing the microcode subroutine call operation, wherein the address identifies an operation generated by decoding a non-microcoded instruction immediately subsequent to the microcoded instruction within the stream of instructions.

IX. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

X. <u>RELATED PROCEEDINGS APPENDIX</u>

There are no related proceedings.